Getting Started: Practice Machine Learning Algorithms in Computer Vision

Yueming Wang

2016. 11. 21

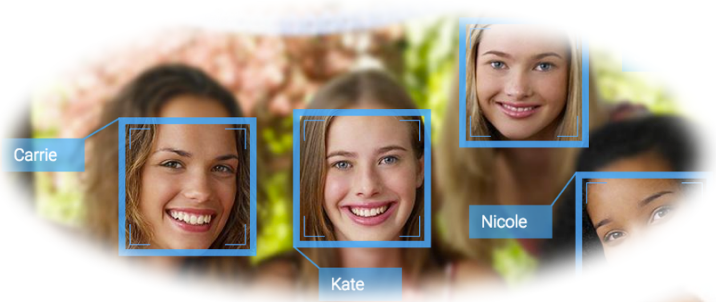
What is Machine Learning

Machine learning

From Wikipedia, the free encyclopedia

For the journal, see Machine Learning (journal).

Machine learning is a subfield of computer science[1] that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.[1] Machine learning explores the construction and study of algorithms that can learn from and make predictions on data.[2] Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions,[3]:2 rather than following strictly static program instructions.



- Database mining
  Large datasets from growth of automation/web.
  E.g., Web click data, medical records, biology, engineering

- Self-customizing programs
  E.g., Amazon, Netflix product recommendations

- Applications can't program by hand.
  E.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.

- Understanding human learning (brain, real AI).

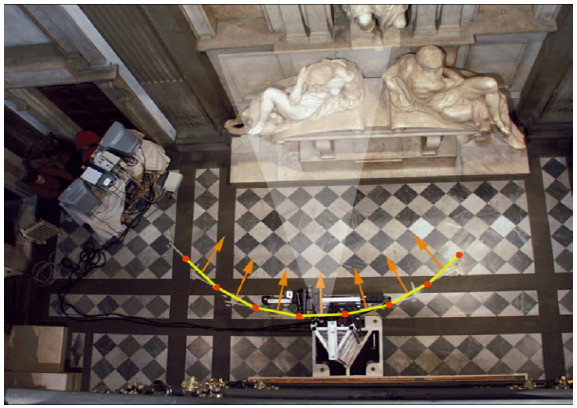
What is Computer Vision

Computer Vision is the study of analysis of pictures and videos in order to achieve results similar to those as by men.

Goal of computer vision is to write computer programs that can interpret images

- Low-level vision
  - Camera models
  - Radiometric measurement
  - Color
- Mid-level vision
  - 3D reconstruction
  - Motion estimation
- High-level vision
  - Segmentation
  - Classifier
  - Image understanding

Image Inpainting



Mid-level Vision

Object recognition

Computer Vision - Examples

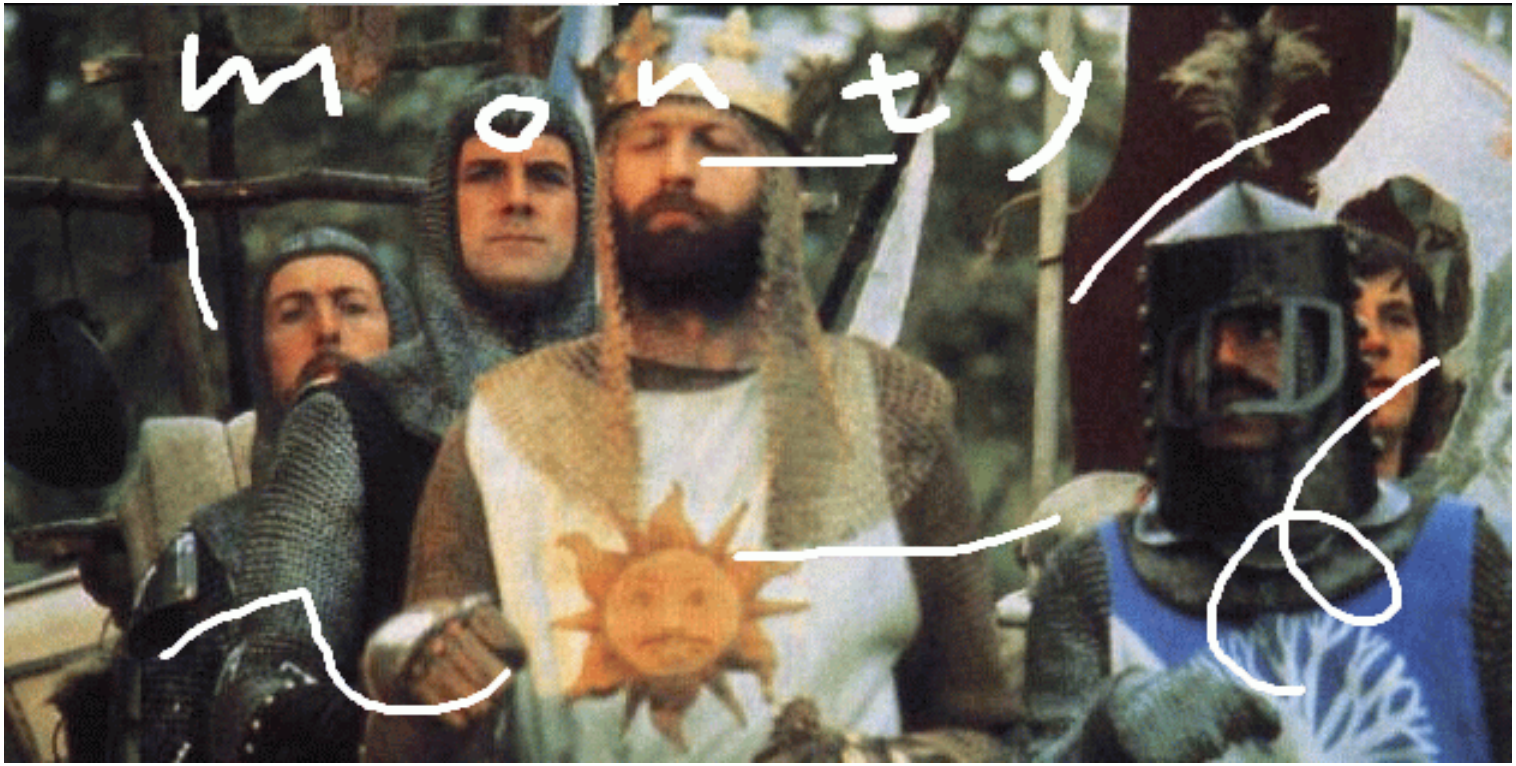
Image Inpainting, M. Bertalmío

Computer Vision - Examples

Image Inpainting, M. Bertalmío

Computer Vision - Examples

Image Inpainting, M. Bertalmío

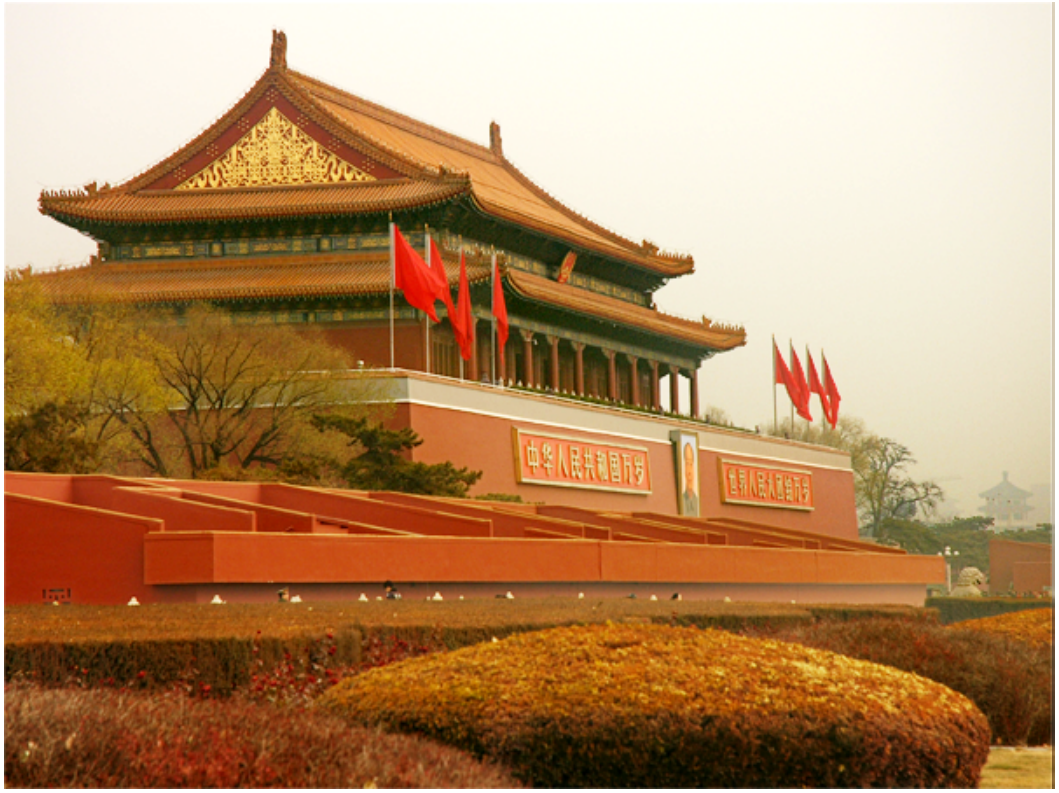


Computer Vision - Examples

去雾， *Dehaze, Microsoft Research Asia, K. He.*

Computer Vision - Examples

Dehaze, Microsoft Research Asia, K. He.

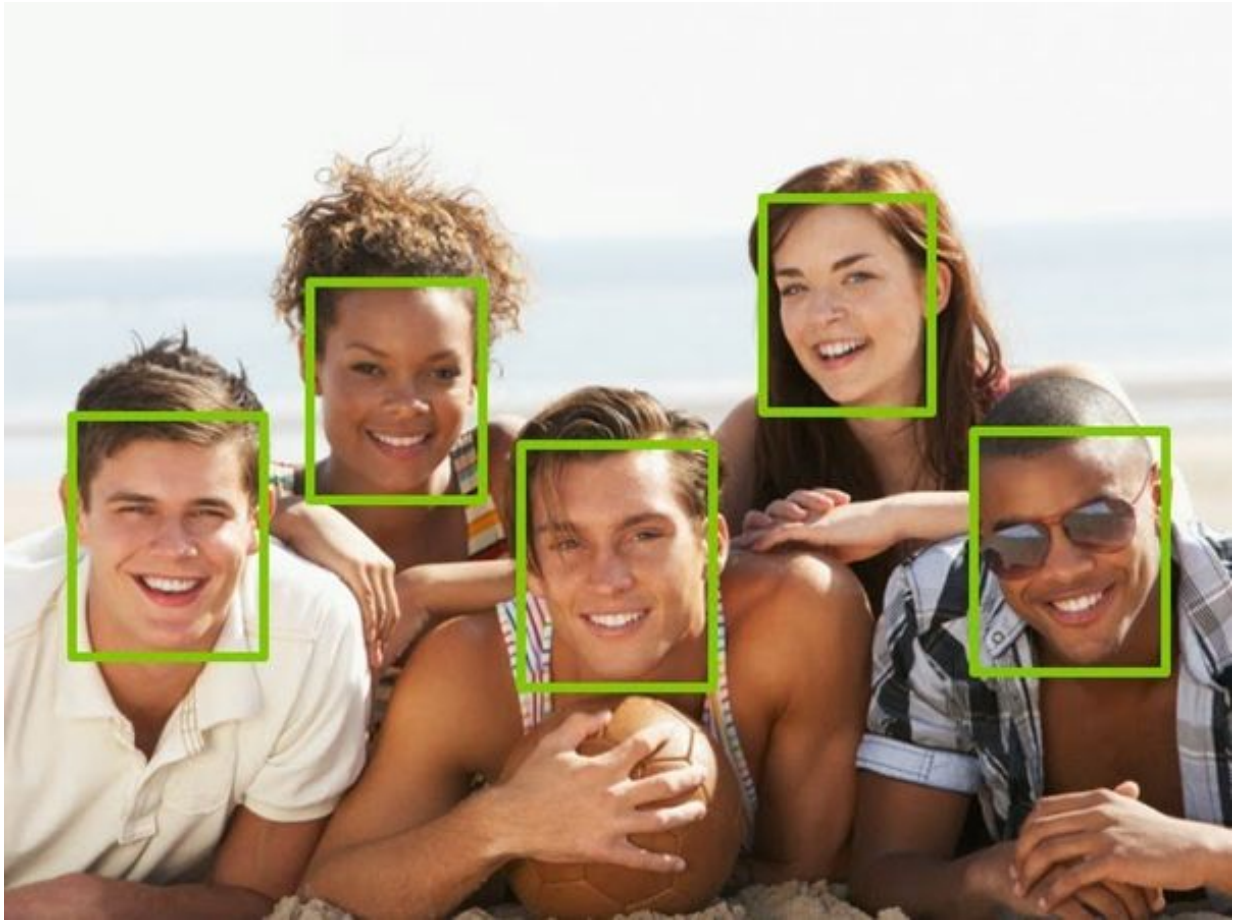


Computer Vision - Examples

Dehaze, Microsoft Research Asia, K. He.

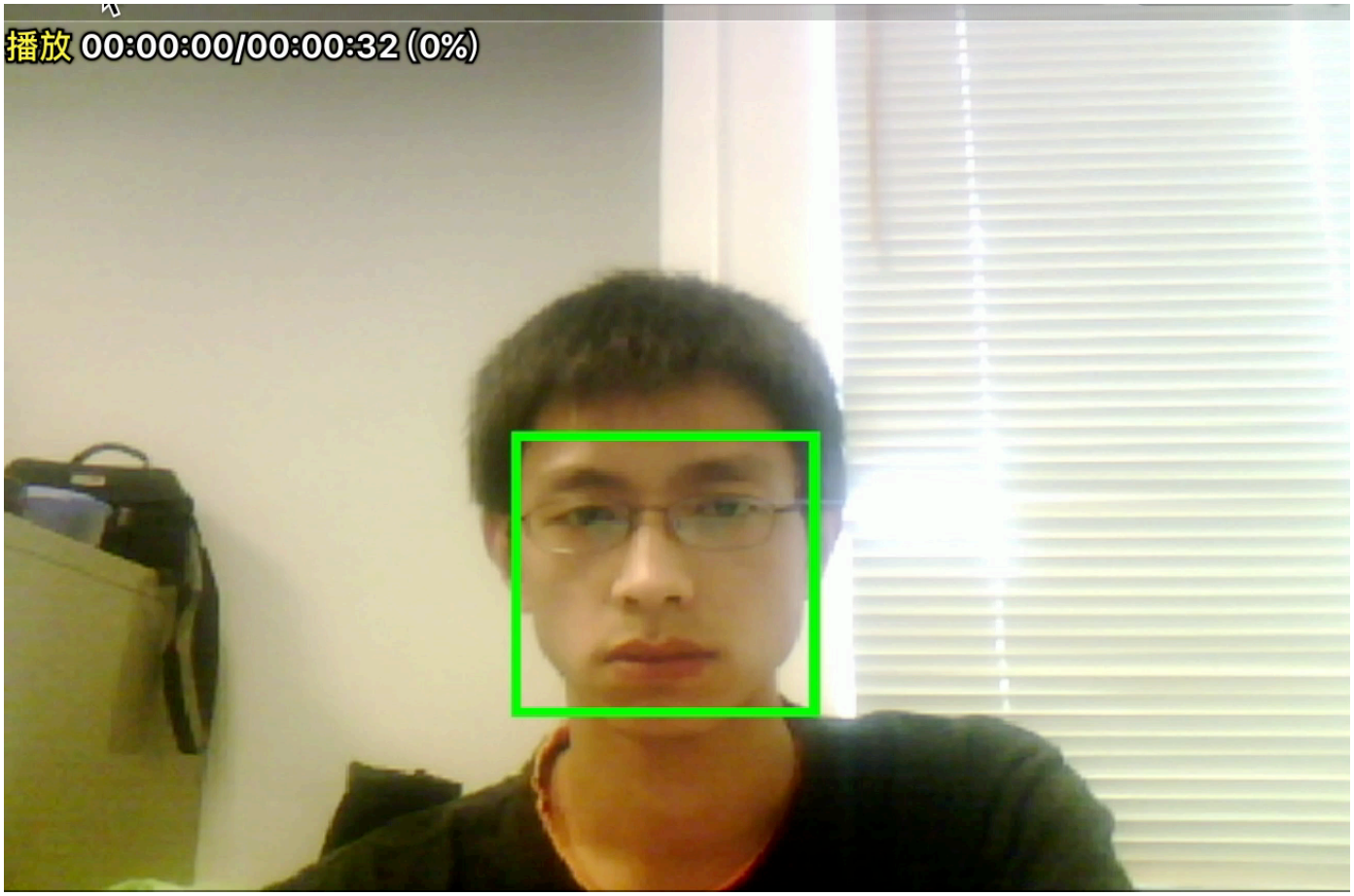
Computer Vision - Examples

Face Detection



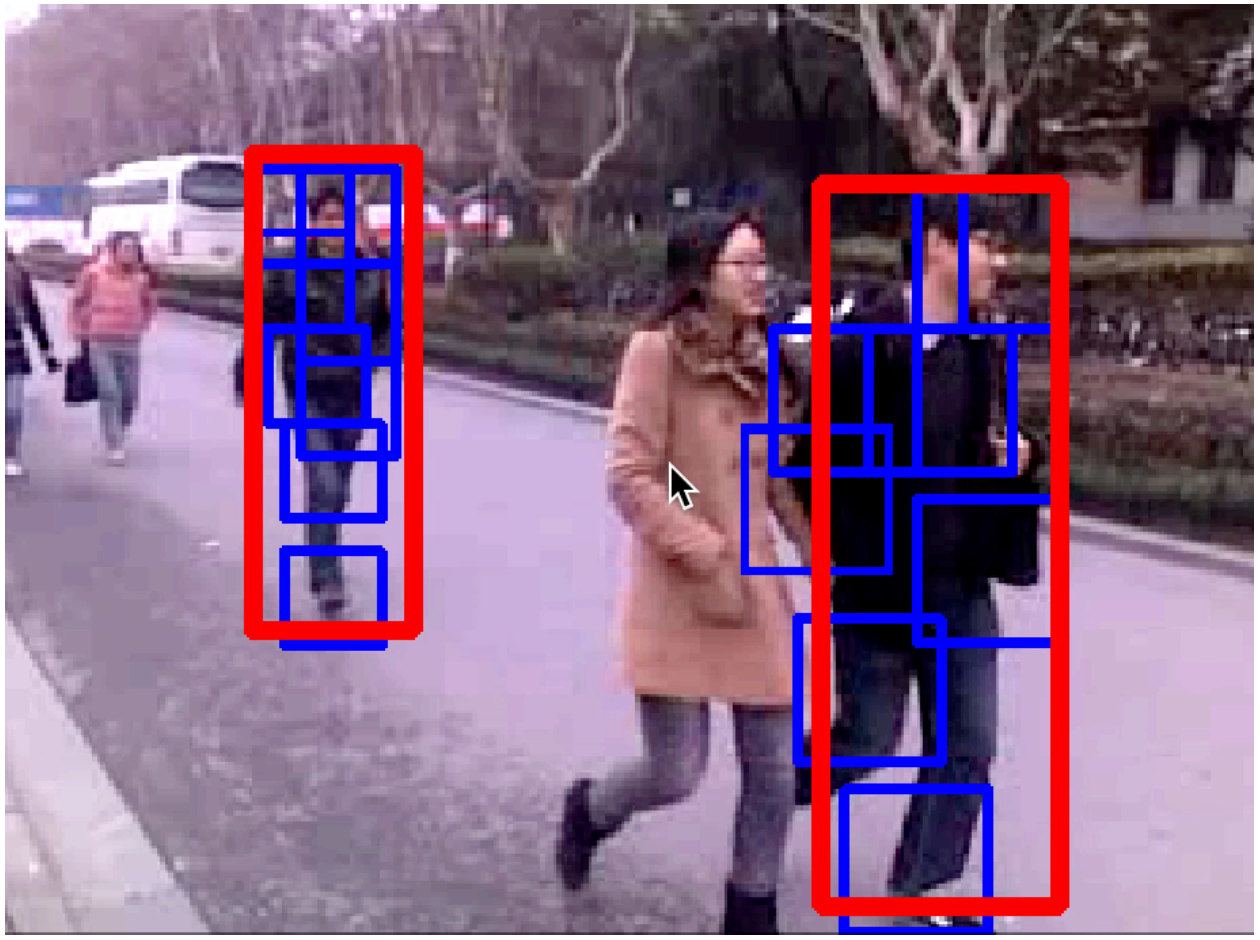
Computer Vision - Examples

Face Detection



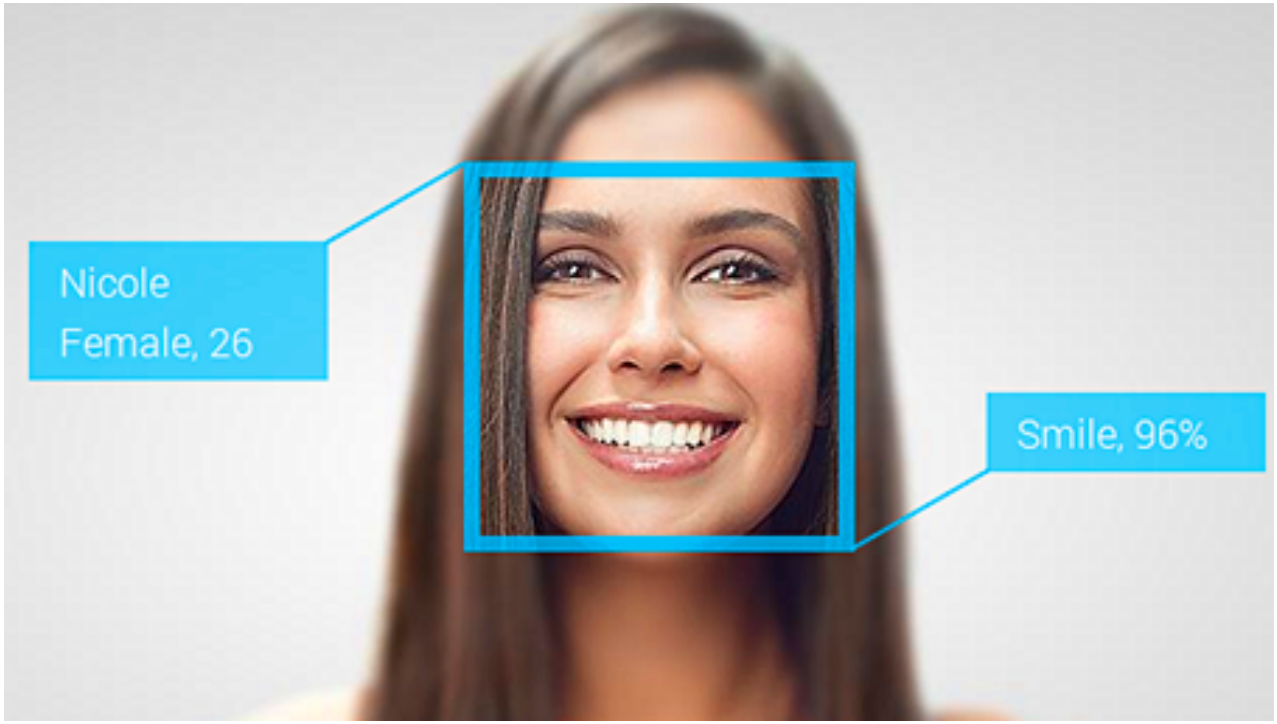
Computer Vision - Examples

Pedestrian Detection



Computer Vision - Examples

Face Recognition



Relationship between ML and CV

- Machine learning focuses more than a model itself, has more mathematics, emphasizes the understanding of models and algorithms
- Computer vision can be seen as an application field, and seems more like problem(s).
- Obviously, the models in ML can be applied to solve CV problems. Indeed, people are doing this.
- Can ML researchers address all problems in CV? NO
- Computer vision has its own characteristics.
  - Data are usually large
  - Processing skills of image and video data are varied
  - Many problems have priors
  - Requiring good programming skills.

ML is Hot partly because of the success of deep learning

Image classification

IMAGENET **Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)**

Team name	Filename	Error (5 guesses)
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422
ISI	pred_FVs_wLACs_weighted.txt	0.26172

http://www.image-net.org/

ML is Hot partly because of the success of deep learning

Image classification

IMAGENET **Large Scale Visual Recognition Challenge 2013 (ILSVRC2013)**

Team name	Comment	Error
Clarifai	Multiple models trained on the original data plus an additional model trained on 5000 categories.	0.11197
Clarifai	Multiple models trained on the original data plus an additional model trained on other 1000 category data.	0.11537
Clarifai	Average of multiple models on original training data.	0.11743
Clarifai	Another attempt at multiple models on original training data.	0.1215
Clarifai	Single model trained on original data.	0.12535
NUS	adaptive non-parametric rectification of all outputs from CNNs and refined PASCAL VOC12 winning solution, with further retraining on the validation set.	0.12953
NUS	adaptive non-parametric rectification of all outputs from CNNs and refined PASCAL VOC12 winning solution.	0.13303
ZF	5 models (4 different architectrues) trained on original data.	0.13511
Andrew Howard	This is an ensemble of convolutional neural networks combining multiple transformations for training and testing and models operating at different resolutions.	0.13555
Andrew Howard	This method explores re weighting the predictions from different data transformation and ensemble members in the previous submission.	0.13564

ML is Hot partly because of the success of deep learning

Image classification

IMAGENET Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)

Team name	Entry description	**Localization error**	Classification error
VGG	a combination of multiple ConvNets (by averaging)	0.253231	0.07405
VGG	a combination of multiple ConvNets (fusion weights learnt on the validation set)	0.253501	0.07407
VGG	a combination of multiple ConvNets, including a net trained on images of different size (fusion done by averaging); detected boxes were not updated	0.255431	0.07337
VGG	a combination of multiple ConvNets, including a net trained on images of different size (fusion weights learnt on the validation set); detected boxes were not updated	0.256167	0.07325
GoogLeNet	Model with localization ~26% top5 val error.	0.264414	0.14828
GoogLeNet	Model with localization ~26% top5 val error, limiting number of classes.	0.264425	0.12724
VGG	a single ConvNet (13 convolutional and 3 fully-connected layers)	0.267184	0.08434
SYSU_Vision	We compared the class-specific localization accuracy of solution 1 and solution 2 by the validation set. Then we chosen better solution on each class based on the accuracy. General speaking, solution 2 outformed solution 1 when there were multiple objects in the image or the objects are relatively small.	0.31899	0.14446
MIL	5 top instances predicted using FV-CNN	0.337414	0.20734
MIL	5 top instances predicted using FV-CNN + class specific window size rejection. Flipped training images are added.	0.33843	0.21023

ML is Hot partly because of the success of deep learning

IMAGENET Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)

MSRA	Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun	We train neural networks with depth of over 150 layers. We propose a "deep residual learning" framework [a] that eases the optimization and convergence of extremely deep networks. Our "deep residual nets" enjoy accuracy gains when the networks are substantially deeper than those used previously. Such accuracy gains are not witnessed for many common networks when going deeper. Our localization and detection systems are based on deep residual nets and the "Faster R-CNN" system in our NIPS paper [b]. The extremely deep representations generalize well, and greatly improve the results of the Faster R-CNN system. Furthermore, we show that the region proposal network (RPN) in [b] is a generic framework and performs excellent for localization. We only use the ImageNet main competition data. We do not use the Scene/VID data.

Ordered by localization error

Team name	Entry description	**Localization error**	Classification error
MSRA	Ensemble A for classification and localization.	0.090178	0.03567
MSRA	Ensemble B for classification and localization.	0.090801	0.03567
MSRA	Ensemble C for classification and localization.	0.092108	0.0369
Trimps-Soushen	combined 12 models	0.122907	0.04649

ML is Hot partly because of the success of deep learning

IMAGENET **Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)**

MSRA	Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun	We train neural networks with depth of over 150 layers. We propose a "deep residual learning" framework [a] that eases the optimization and convergence of extremely deep networks. Our "deep residual nets" enjoy accuracy gains when the networks are substantially deeper than those used previously. Such accuracy gains are not witnessed for many common networks when going deeper. Our localization and detection systems are based on deep residual nets and the "Faster R-CNN" system in our NIPS paper [b]. The extremely deep representations generalize well, and greatly improve the results of the Faster R-CNN system. Furthermore, we show that the region proposal network (RPN) in [b] is a generic framework and performs excellent for localization. We only use the ImageNet main competition data. We do not use the Scene/VID data.

ML is Hot partly because of the success of deep learning

IMAGENET **Large Scale Visual Recognition Challenge 2016 (ILSVRC2016)**

Object detection (DET)[top]

Task 1a: Object detection with provided training data

Ordered by number of categories won

Team name	Entry description	**Number of object categories won**	mean AP
CUImage	Ensemble of 6 models using provided data	109	0.662751

Object localization (LOC)[top]

Task 2a: Classification+localization with provided training data

Ordered by localization error

Team name	Entry description	**Localization error**	Classification error
Trimps-Soushen	Ensemble 3	0.077087	0.02991

ML is Hot partly because of the success of deep learning

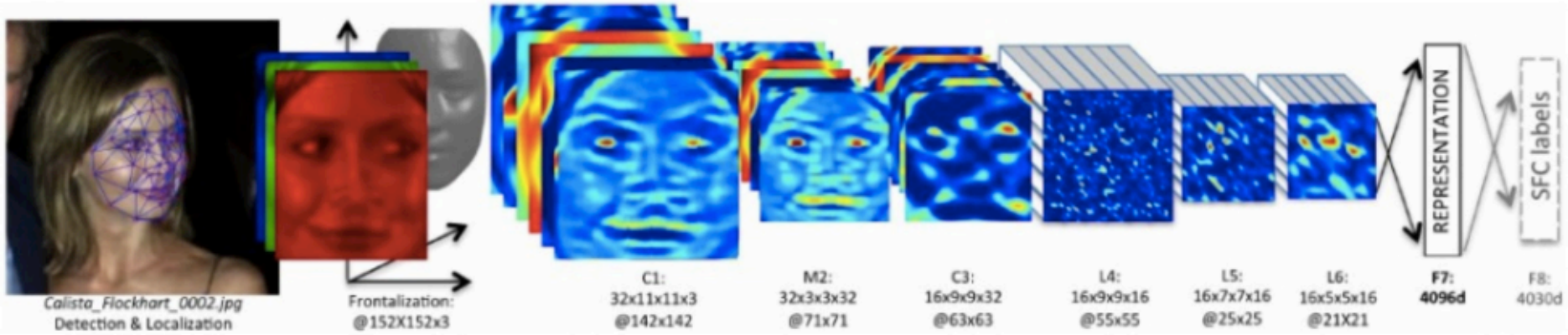
IMAGENET Large Scale Visual Recognition Challenge 2016 (ILSVRC2016)

CUImage	Hongsheng Li*, Kai Kang* (* indicates equal contribution), Wanli Ouyang, Junjie Yan, Tong Xiao, Xingyu Zeng, Kun Wang, Xihui Liu, Qi Chu, Junming Fan, Yucong Zhou, Yu Liu, Ruohui Wang, Shengen Yan, Dahua Lin, Xiaogang Wang	Compared with CUImage submission in ILSVRC 2015, the new components are as follows. (1) The models are pretrained for 1000-class object detection task using the approach in [a] but adapted to the fast-RCNN for faster detection speed. (2) The region proposal is obtained using the improved version of CRAFT in [b]. (3) A GBD network [c] with 269 layers is fine-tuned on 200 detection classes with the gated bidirectional network (GBD-Net), which passes messages between features from different support regions during both feature learning and feature extraction. The GBD-Net is found to bring ~3% mAP improvement on the baseline 269 model and ~5% mAP improvement on the Batch normalized GoogleNet.

ML is Hot partly because of the success of deep learning

Face recognition

Deepface

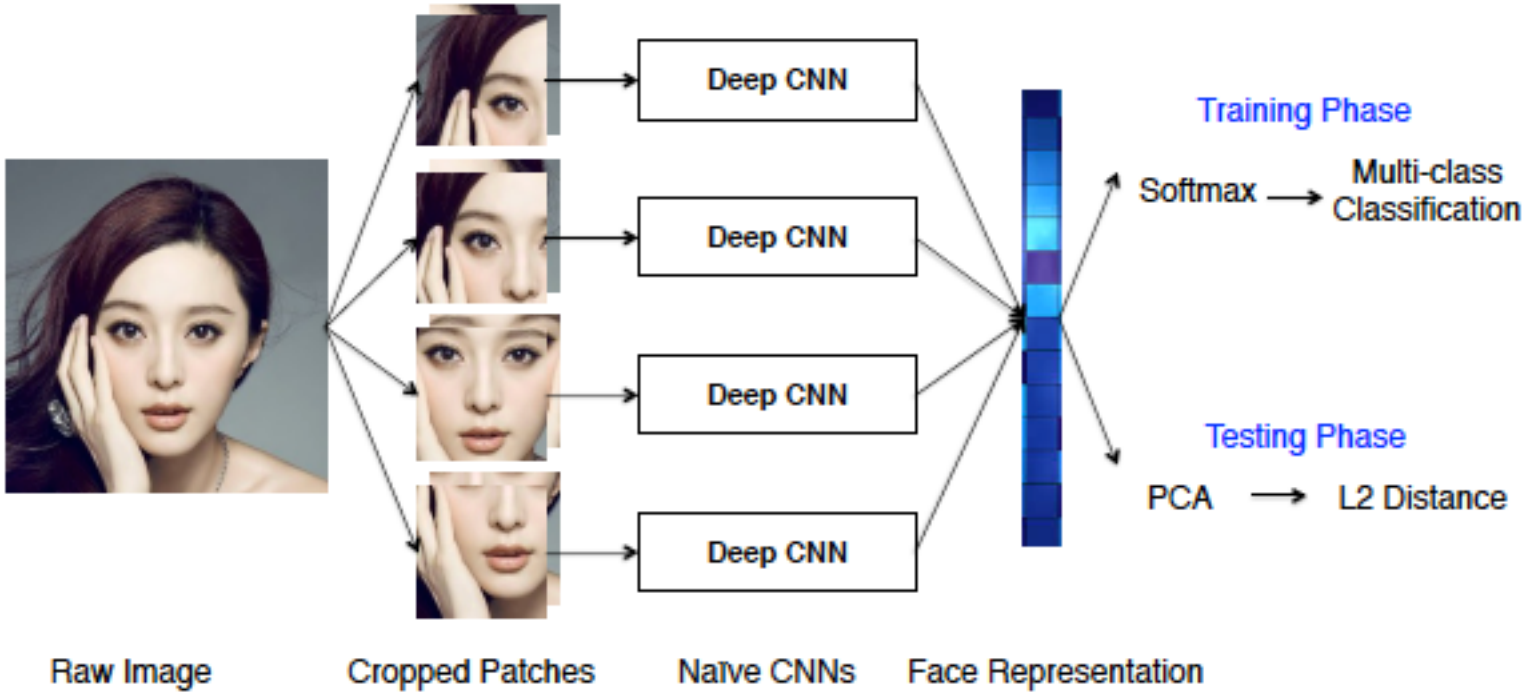


simple classifier. Our method reaches an accuracy of 97.35% on the Labeled Faces in the Wild (LFW) dataset, reducing the error of the current state of the art by more than 27%, closely approaching human-level performance.

Taigman, Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014.

ML is Hot partly because of the success of deep learning

Face recognition



E. Zhou, Z. Cao, Q. Yin, Naive-Deep Face Recognition: Touching the Limit of LFW Benchmark or Not? arXiv, 2015

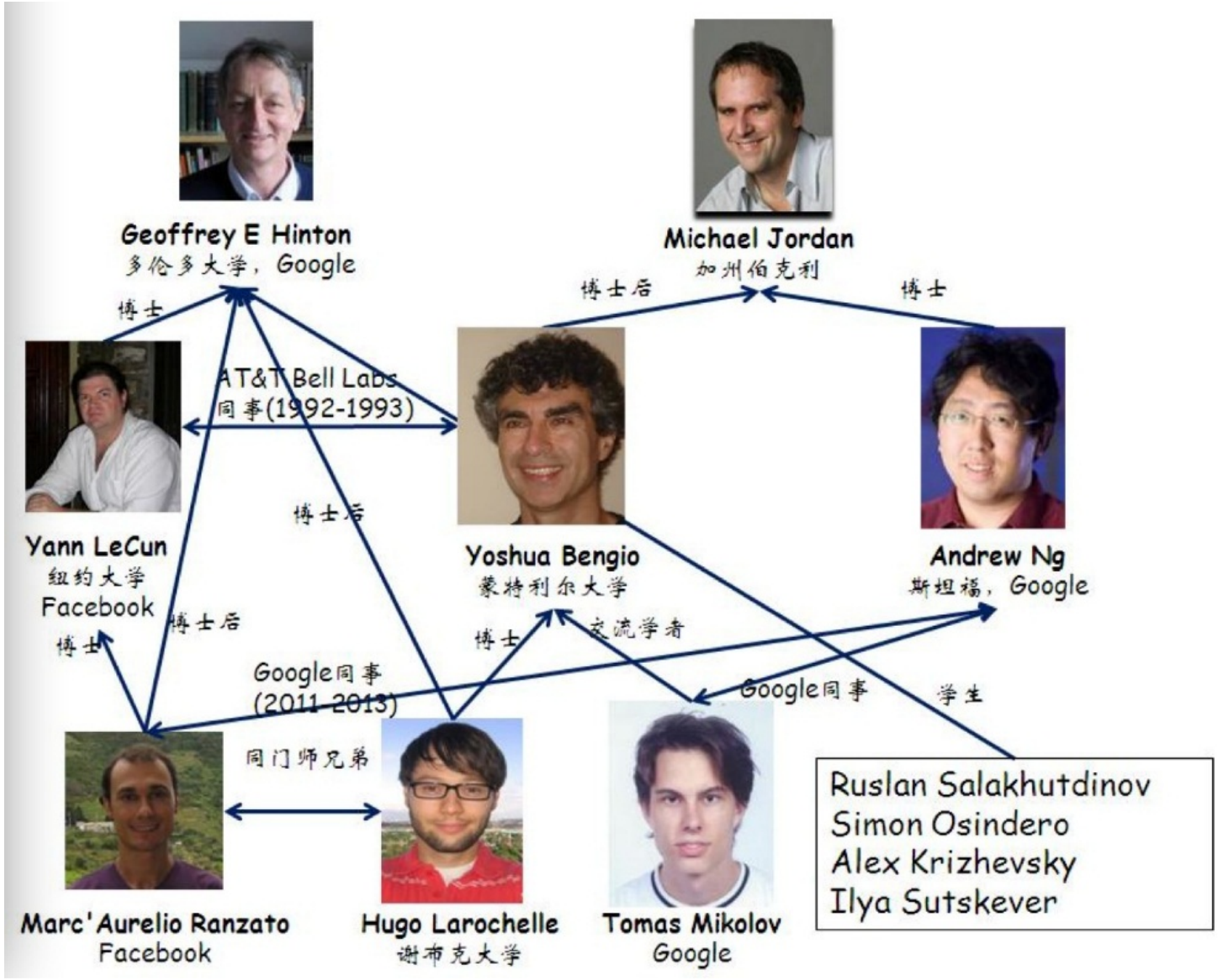
ML is Hot partly because of the success of deep learning

Pose estimation

Deeppose

Toshev, Alexander, and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks." Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014.

Who is who in deep learning?



Machine Learning Open Courses

coursera


Andrew Ng

https://www.coursera.org/learn/machine-learning/

http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning

http://cs229.stanford.edu/

https://www.cs.toronto.edu/~hinton/csc2535/lectures.html


Geoffrey E Hinton
多伦多大学，Google

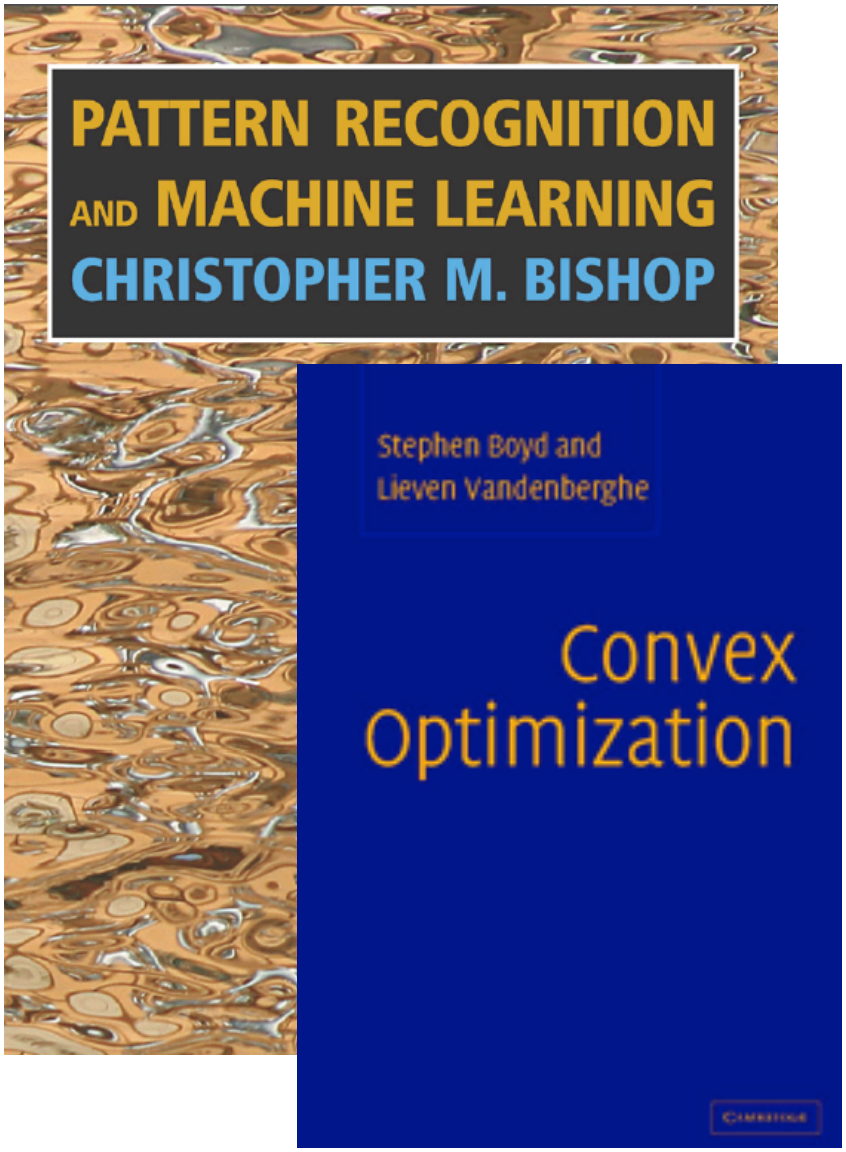

Yann LeCun
纽约大学

http://www.cs.nyu.edu/~yann/2010f-G22-2565-001/index.html

http://www.cs.berkeley.edu/~jordan/courses/294-fall09/


Michael I. Jordan

Machine Learning Books



2 Probability Distributions

3 Linear Models for Regression

4 Linear Models for Classification

5 Neural Networks

6 Kernel Methods

7 Sparse Kernel Machines

8 Graphical Models

9 Mixture Models and EM

10 Approximate Inference

11 Sampling Methods

12 Continuous Latent Variables

13 Sequential Data

14 Combining Models

- Know everything: God
- Be able to derive all formula, prove every identity: Mathematician
- Simply find the existing code of the algorithms and run it: basic computer engineers
- WE:
  - Know what are the problems in applications
  - Understand the mechanisms and strength of the algorithms. It's better if derivation is good. Not necessary to prove everything.
  - We can research, program, and debug !!!

Target & Content of this Course

- Neither a full ML course nor a full CV course
- This is a short project course. The purpose is to train the new graduate students to quickly go into the research and development of ML and PR. We will begin with a computer vision problem, and end with a demo. Understanding the mechanisms and strengths of ML algorithms and programming (coding) are two highlights in the whole procedure. Besides, you will be required to:
  - Set up your own scientific programming environment
  - Prepare the data by yourself
  - Quickly start to program with unfamiliar languages and packages
  - Quickly solve your problems in coding by "correctly" using search
  - Find the reasons of occurred problems in ML algorithms and solve them
  - Set up the experiments and standard protocols for evaluation
  - Develop a prototype
  - …
- Content covers,
  - Image classification, logistic regression, SVM, and Convolutional Neural Network (so-called deep learning)

Now, we go to the problem to be addressed!!

Problem: How to Find Basketball Goals in a video

More precisely, given a video, how to determine the moment(s) (or frames) that a goal occurs?

Analysis

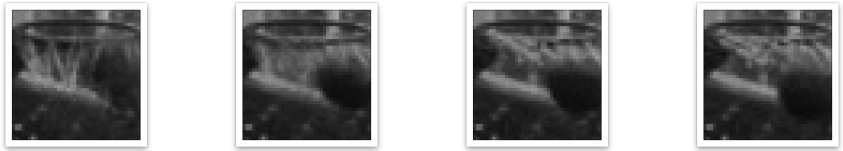
- One goal can involve a sequence of frames
- The hoop nearby region contains important features
- Idea1: distinguish hoop-with-ball frames from hoop-without-ball frames, then merge the continuous hoop-with-ball frames as a goal?

Possible

This first idea

- Idea1: distinguish hoop-with-ball frames from hoop-without-ball frames, then merge the continuous hoop-with-ball frames as a goal?

1 What region to classify? How to prepare data for machine learning?
label the hoop region and crop it for each frame and annotate each frame as either hoop-with-ball or hoop-without-ball. After that, we have two small image sets. One is positive sample set and the other is negativ One.

positive sample set 

negative sample set 

2 Then, the problem becomes how to learn a classifier to determine whether an input small picture is positive (hoop-with-ball) or hoop-without-ball.

This is a standard machine learning problem

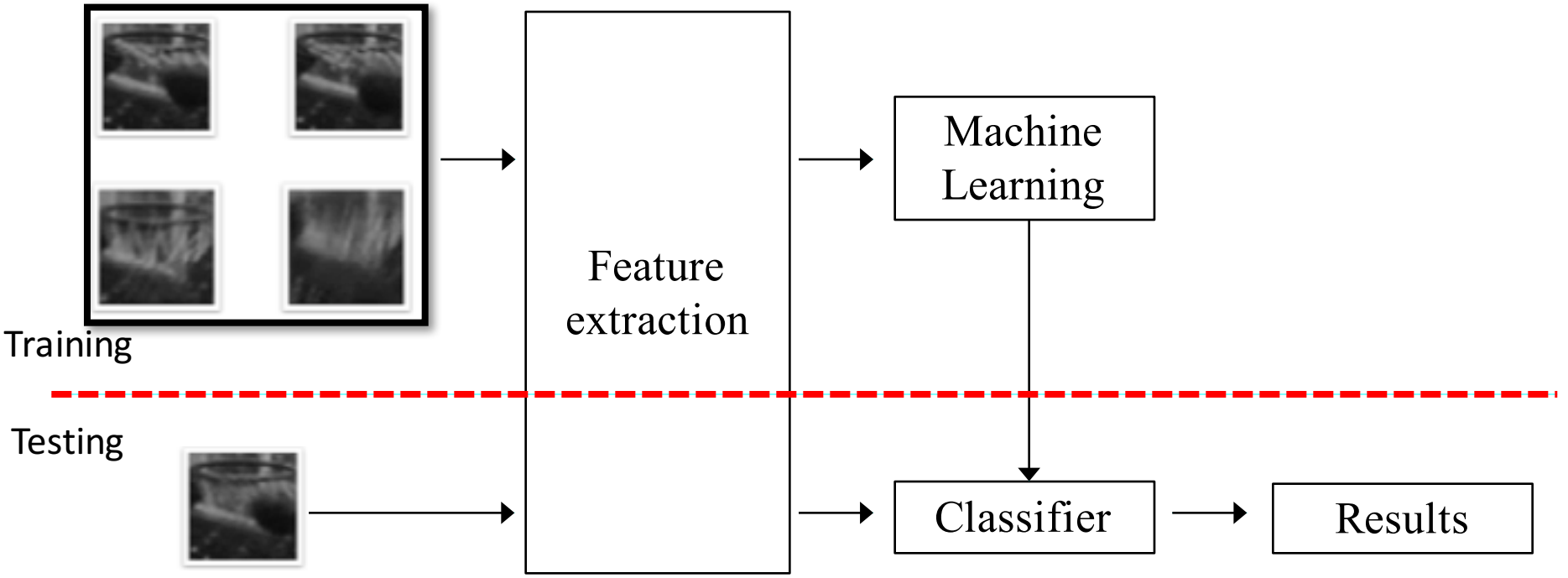
3 According the output of the classifier, merge the continuous positive frames to be a goal

More in step 2

- how to learn a classifier to determine whether an input small picture is positive (hoop-with-ball) or hoop-without-ball

This is a standard machine learning problem

usually, we will split the sample set to training and testing set (sometimes an additional validation set). Then, we begin learn …



OK, let’s implement it step by step …

Setting up Your Scientific Python (Mac OS X)

Yueming Wang

2015/7/5

This document shows how to set up a scientific python environment on your own computer. The examples are mainly for Mac OS X because I am using this operating system, ☺. However, usually we need almost the same components for other OSs. If you understand the framework for a scientific python stack, it is easy to find the detail of instructions on other OSs from Internet.

I will show how to install python and basic scientific python stack as follows,

- gcc (or vc++) and git (development headers)

  A C++ complier is needed for some packages communicating between C++ and python.

  Git s a free and open source distributed version control system.

  Url: https://developer.apple.com/xcode/downloads/

For Mac OS X users, I recommend to install Xcode from the apple APP store. Once it is installed, then you already have the tools.

- Homebrew (for Mac OS X) or apt-get (for Linux)

  Homebrew is a general package manager for OS X. The counterpart in Linux is apt-get.

To install Homebrew, paste the following in a terminal:

```
ruby -e "$(curl -fsSL https://raw.github.com/mxcl/homebrew/go)"
```

- Python2.x or Python 3.x

  The python 2 series and 3 series are different in not only the version number. They have different grammars, packages, and are not compatible with each other. Python 2.x is more popular, so I recommend it. The latest version is Python 2.7.10.

  Url: https://www.python.org/

Now you can use brew to install Python:

```
brew install python
```

Note that it has been installed to "/usr/local/Cellar/python/2.7.9/". However, the terminal executes the python in "/usr/local/bin/python", which is a link pointing to the system default python versions "/System/Library/Frameworks/Python.framework/Versions/Current/bin/". So if we run the commands:

```
Python
```

and it shows the information:

```
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
```

But if you type:

```
/usr/local/Cellar/python/2.7.9/bin/python
```

It shows:

```
Python 2.7.9 (default, Jul  5 2015, 16:18:17)
```

So how to set the default python version? The answer is to change the file ~/.bashrc. Open it and add an alias for "python". (when the current user run bash shell, read it first. The other related file is ~/.bash_profile, which loading the configure only once when the user login.)

```
s python="/usr/local/Cellar/python/2.7.9/bin/python"
```

Restart the terminal to make it effective.

Then, you have successfully installed the python interpreter. The next step is to set your python path, so that the packages can be installed to that path in future and the python can find dependencies from the path . In a terminal,

```
vim ~/.bash_profile
```

and add the following to the end of the file:

```
export PYTHONPATH="$PYTHONPATH:/usr/local/lib/python2.7/site-packages"
```

At this point you should close your terminal and open a new one so that this PATH setting is in effect for the rest of the installation.

Thus, usually the following packages should be in the python path in ~/.bash_profile, i.e.,

```
"/usr/local/lib/python2.7/site-packages"
```

But sometimes the system has a default version at:

```
"/Library/Python/2.7/site-packages/"
```

- Pip

  Unlike homebrew, it is a tool for installing and managing packages specific for python, such as those found in the Python Package Index.

Usually if you have homebrew installed, pip installs with python. Otherwise, type the following in the terminal:

```
sudo easy_install pip
```

Highlight: PyPI – the Python Package Index: https://pypi.python.org/pypi

- Numpy

  It is the fundamental package for scientific computing with Python, especially as a powerful n-dimensional array container.

  Url: http://www.numpy.org/

  Tutorial*: http://wiki.scipy.org/Tentative_NumPy_Tutorial*

Installation:

```
pip install numpy
```

and testing,

```
python
import numpy
```

```
# If you have installed numpy in early installation, you can upgrade it by:
pip install --upgrade numpy
```

```
# If you want to upgrade all packages installed by pip at one time, type the following python code:
import pip
from subprocess import call
for dist in pip.get_installed_distributions():
    call("pip install --upgrade " + dist.project_name, shell=True)
```

- SciPy

  SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. It is a undamental library for scientific computing.

  Url: http://www.scipy.org/

Installation:

```
pip install Scipy
```

and run

```
python
import Scipy
```

- Matplotlib

  It is a tool for comprehensive 2D Plotting.

  Url: http://matplotlib.org/

Installation:

```
pip install Matplotlib
```

and run

```
python
import Matplotlib
```

- Opencv

  It is an open library for computer vision research and applications, containing many CV algorithms and having more than 47 thousand users.

  Url: http://opencv.org/

Installation:

1. Installing FFmpeg

FFmpeg is used to encode and decode multimedia data. Without it, some functions will not compile properly. Run the following line in the terminal:

```
brew install ffmpeg
```

2. Installing Cmake

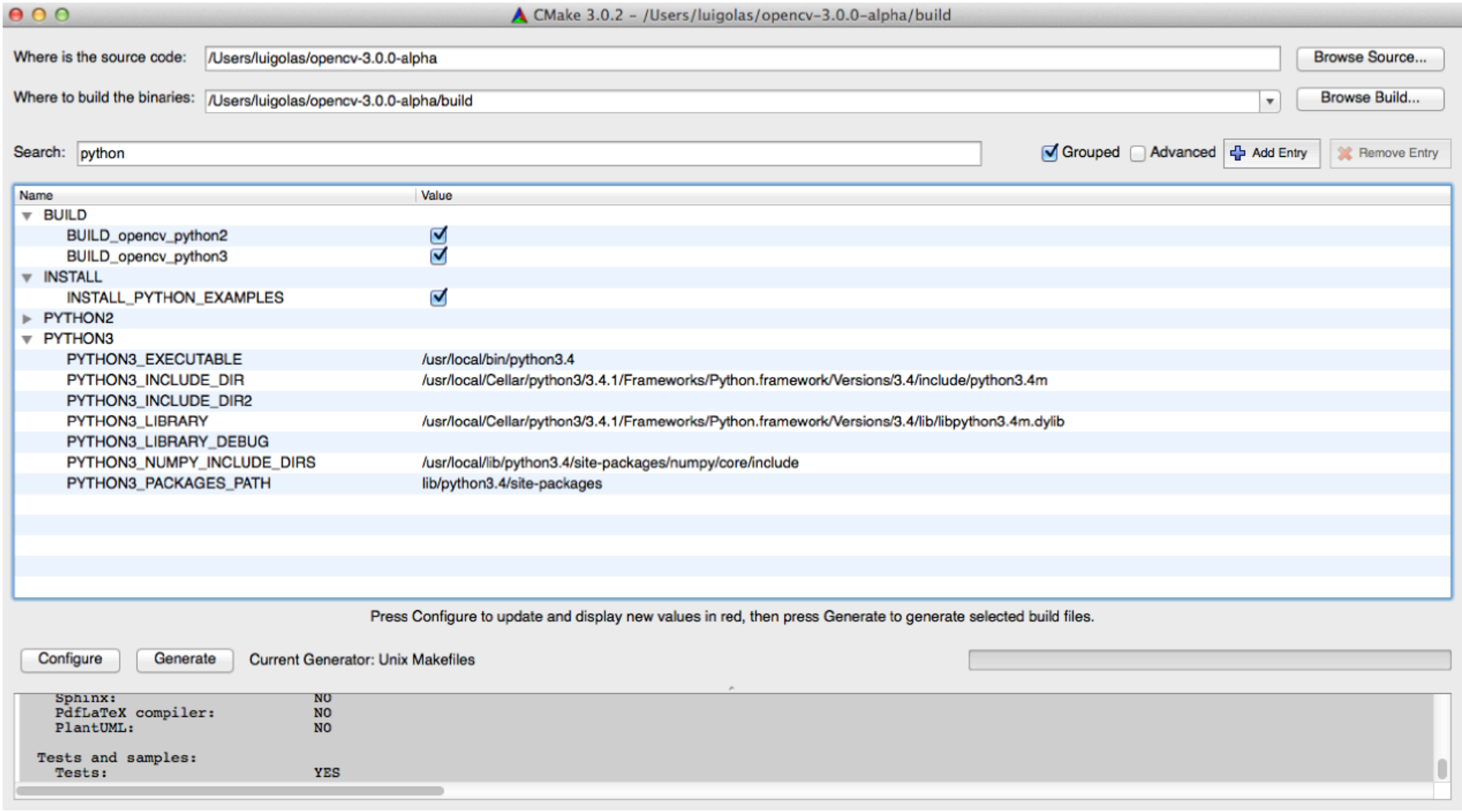
There is a GUI if you download it from its homepage: http://www.cmake.org/download/

3. installing opencv3.0

First download it from http://opencv.org/downloads.html. Unzip it. Then it is the time to use Cmake. Open it and set where the source code is. And make a build folder inside the opencv3.0 folder. Click 'configure' and search 'python'. Set the configuration as the figure shows. Now, what I explained in the previous installation about python shows its power. Without it, setting would be a difficult problem. But at this moment, it is NOT. Note that the Current Generator is Unix Makefiles.

After everything is OK, click "Configure". If everything goes fine, click generate. Again, if there is no error, open a terminal and go to the "build" folder, something like:

Opencv


CMake 3.0.2 - /Users/luigolas/opencv-3.0.0-alpha/build
Where is the source code: /Users/luigolas/opencv-3.0.0-alpha
Browse Source...
Where to build the binaries: /Users/luigolas/opencv-3.0.0-alpha/build
Browse Build...
Search: python
Grouped
Advanced
Add Entry
Remove Entry
Name
Value
BUILD
BUILD_opencv_python2
BUILD_opencv_python3
INSTALL
INSTALL_PYTHON_EXAMPLES
PYTHON2
PYTHON3
PYTHON3_EXECUTABLE /usr/local/bin/python3.4
PYTHON3_INCLUDE_DIR /usr/local/Cellar/python3/3.4.1/Frameworks/Python.framework/Versions/3.4/include/python3.4m
PYTHON3_INCLUDE_DIR2
PYTHON3_LIBRARY /usr/local/Cellar/python3/3.4.1/Frameworks/Python.framework/Versions/3.4/lib/libpython3.4m.dylib
PYTHON3_LIBRARY_DEBUG
PYTHON3_NUMPY_INCLUDE_DIRS /usr/local/lib/python3.4/site-packages/numpy/core/include
PYTHON3_PACKAGES_PATH lib/python3.4/site-packages
Press Configure to update and display new values in red, then press Generate to generate selected build files.
Configure
Generate
Current Generator: Unix Makefiles
Sphinx: NO
PdfLaTeX compiler: NO
PlantUML: NO
Tests and samples:
Tests: YES

Opencv

After everything is OK, click “Configure”. If everything goes fine, click generate. Again, if there is no error, open a terminal and go to the “build” folder, something like:

```
cd …/opencv-3.0/build
```

Now execute the make command:

```
make –jx
```

where x is the number of cores of your CPU. The next step is,

```
sudo make install
```

If there is no error, try it in python

```
python
Import cv2
```

- Theano

  A library, developed by LISA, makes it easier to implement deep learning algorithms.

  Url:

Installation:

```
pip install Theano
```

after installed, testing your installation

```
python
import theano
theano.test()
```

- IDE: Pycharm

Finally, you need an integrated Development Environment (IDE) for programming in python. I am using Pycharm. The Community Edition is free. So I use it.

https://www.jetbrains.com/pycharm/download/

- Anaconda (all in one)

  An all-in-one python distribution containing more than 330 of the most popular Python packages for science, math, engineering, data analysis.

  Url: https://store.continuum.io/cshop/anaconda/

To be continued …

- Others, including CUDA, BLAS, Boost, and IO library hdf5

Back to the first idea

- Preparing data (step 1)

  Label the hoop region and crop it for each frame and annotate each frame as either hoop-with-ball or hoop-without-ball.

Python + OpenCV

Note: it is highly important to consistently label the hoop region and hoop-with-ball frame sequence. The former is for feature alignment. The latter keeps consistent pattern for learning.

- In this step, you need the packages:
  - Numpy: store an image as an array

    Tutorial: http://wiki.scipy.org/Tentative_NumPy_Tutorial
  - OpenCV: load video, retrieval frames, show images, and processing mouse and keyboard events.

    http://docs.opencv.org/opencv2refman.pdf

Preparing data – Label and annotate

1. Packages

import cv2
from numpy **import** *

2. Load videos

cap = cv2.VideoCapture(fn_video)
success, frame = cap.read()

3. Show image

cv2.imshow('img', frame)
cv2.waitKey(0)

4. Mouse callback

cv2.setMouseCallback("Image", onMouse, hoopPos)

5. Draw Rectangle

cv2.rectangle(…)

6. Locate a frame in a video

cap.set(cv2.CAP_PROP_POS_FRAMES, curFrame)

Run code

```
hoopPos = LabelFunc.getHoopPostion(fn_video)
LabelFunc.labelGoalFrames(fn_video, hoopPos, fn_annotation)
```

Preparing data – Crop the hoop regions out

1. Crop from numpy array

img = frame[hoopPos[1]:hoopPos[3], hoopPos[0]:hoopPos[2]]

2. Convert color image to gray

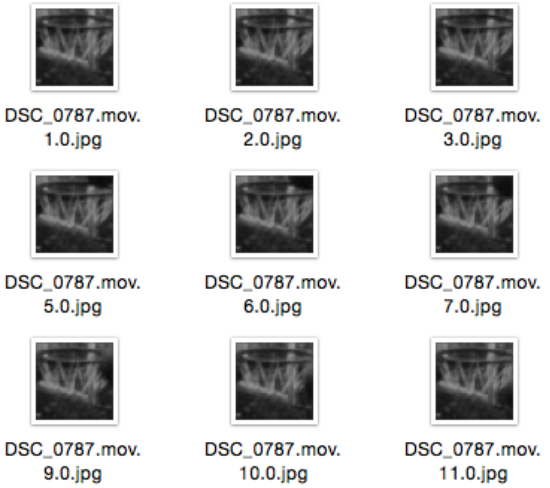
dst = Cv2.cvtColor(src,
cv2. COLOR_BGR2GRAY)

3. Save images

Cv2.imwrite(filename, img)

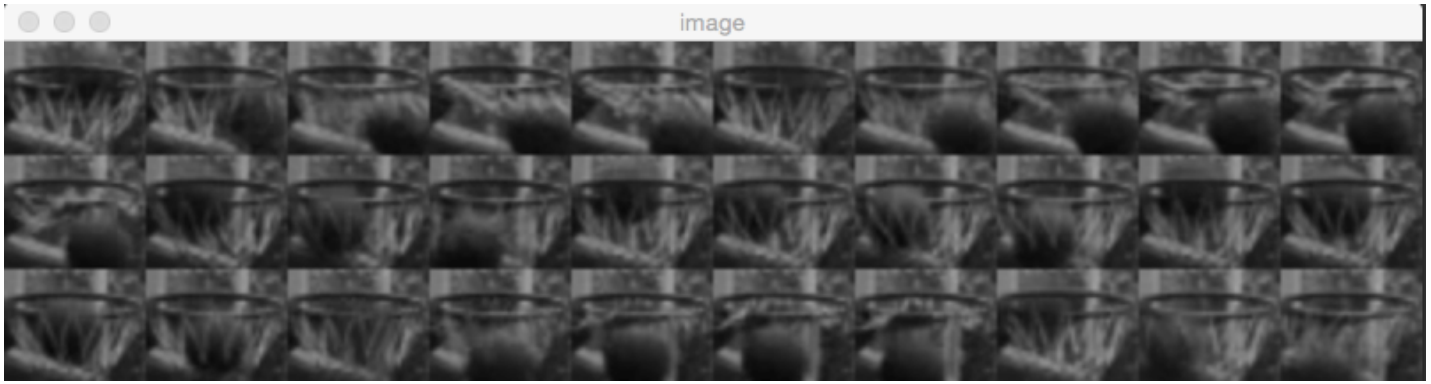
Run code `LabelFunc.CropHoopKeyFrames(fn_video, fn_annotation, pos_out_dir, neg_out_dir, w, h)`

Now, we have the following images:



Preparing data – Carefully consider file organization

- Considering a 1-hour video with fps = 25, there are 25x60x60=90,000 frames. So there are same number of small images. Storing them in one folder leads to
  - Slow IO
  - Even opening the folder to check is difficult
- Since a small picture is not more than 40x40 Bytes = 1.6k, we can put everything to one file, with the size of about 140M. It depends on you how to arrange the small data in one big data structure. An big 3-D array is simple and satisfies our use.
- The drawback is now we can not check the images by opening the folder. Don't worry. Write an program "myExplorer" to explore them.



Preparing data – Carefully consider framework of your code

- In the first attempt, we write some functions and call them in a script to perform annotation. If we have 10 video files, we need to change the code and comment some other code many times during experiments. Further, if we add new functions, the script will become complicate, difficult to read, and easy to bring errors.
- Use a configuration file to maintain all parameters and make your scientific code easier to manage.

```
###### task to be executed, should be one of the following:
###### label / crop / train / crossvalidataion / TestROC
gl_task = label
#################################################################
###### for label
label_fn_video = ../videos/DSC_0787.mov
label_fn_annotation = ../annotations/DSC_0787.txt
#################################################################
##### for crop
crop_fn_video = ../videos/DSC_0787.mov
crop_fn_annotation = ../annotations/DSC_0787.txt
crop_dir_pos = ../possamples/
crop_dir_neg = ../negsamples/
crop_size = 40, 40
#################################################################
##### for training or crossvalidation
#### feature_type: 1 = one frame HoG, 2 = two frames HoG, 3 = thre
#### _algorithm: 1 = SVM, 2 = CNN
train_fn_pos = ../PosSamples/DSC_0790.npy; ../PosSamples/DSC_0787.
train_fn_annotation_pos = ../Annotations/DSC_0790.txt; ../Annotati
```

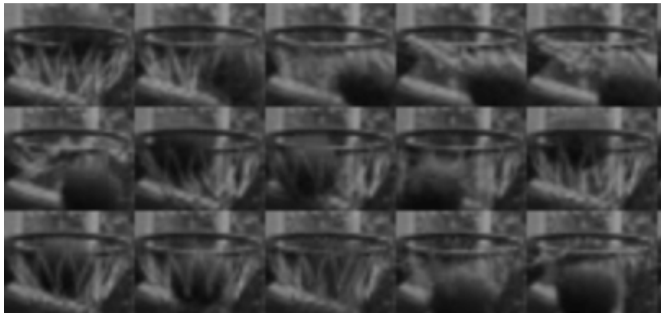
```
print('--------------------')
print("Read the configure file: ")
fp = open('configure.txt', 'r')
strConfig = fp.readlines()
label_vFn = ''
label_vAnnFile = ''
crop_vFn = ''
crop_vAnnFile = ''
outDIRPos = ''
outDIRNeg = ''
task = ''  # label or crop
size_wh = []

for line in strConfig:
    line = line.strip('')
    if line.find('gl_task') >= 0:
        task = line.split('=')[1].strip()
    if line.find('label_fn_video') >= 0:
        label_vFn = line.split('=')[1].strip()
        print(label_vFn)
```

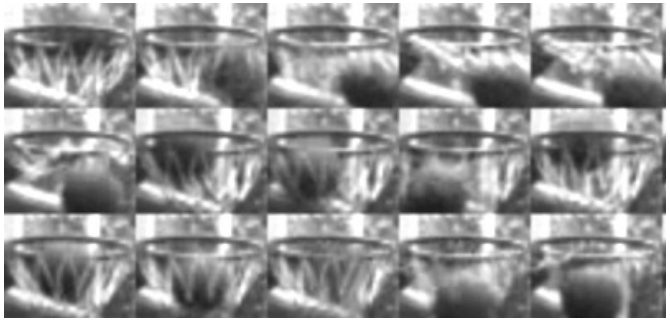
```
print('task = '+task)
if task == 'label':
    hoopPos = LabelFunc.getHoopPostion(label_vFn)
    LabelFunc.labelGoalFrames(label_vFn, hoopPos, label_vAnnFile)
elif task == 'crop':
    LabelFunc.CropHoopKeyFrames(crop_vFn, crop_vAnnFile, outDIRPos
else:
    print("no task is designated.")
```

Feature Extraction

- pixel values as a feature

The simplest choice is to concatenate pixel values to a vector in the order of the row index. Clearly the features are variable for different light conditions.



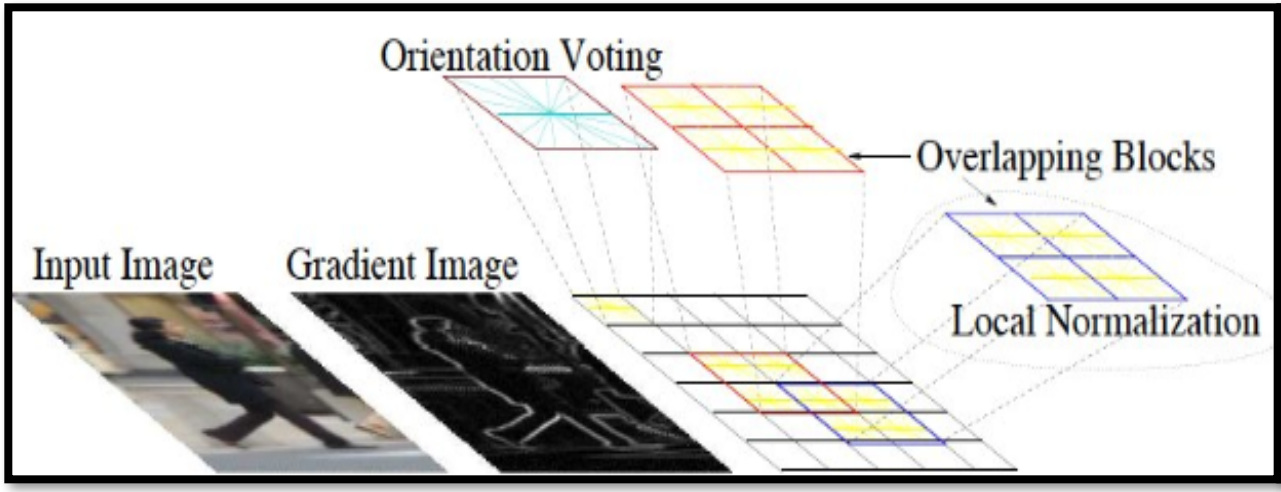


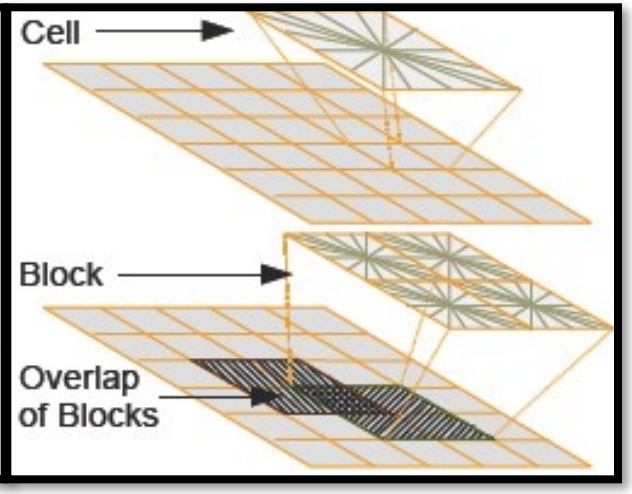
- Observations
  - Shape features and Edge features, which reflects the feature in first-order derivation should be more stable
  - Statistical features should be more robust than the first-order derivations
  - Local patch scheme is important to reflect the geometrical relationship of the objects in images

- Shape + statistical + local
  - SIFT: probably the most famous in CV, (IJCV2004, Google citation: 30602

  - LBP: widely used in face recognition
  - HoG: a variation of SIFT, firstly used for human detection
  - …

Feature: Histogram of Gradient (HoG)

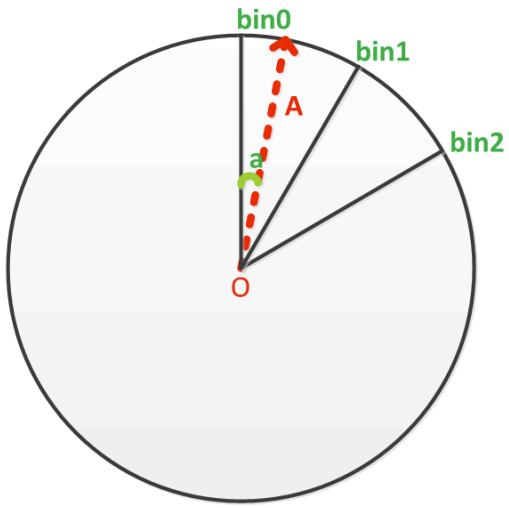


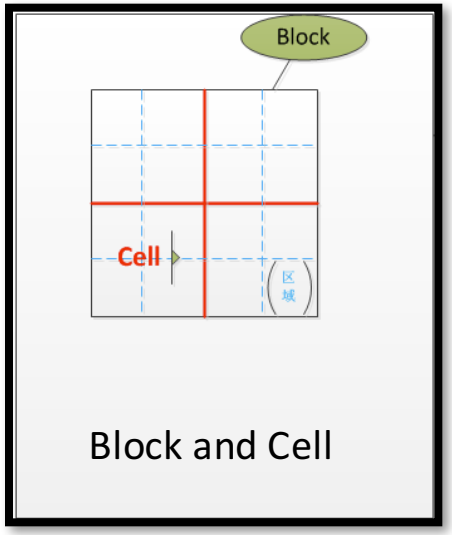


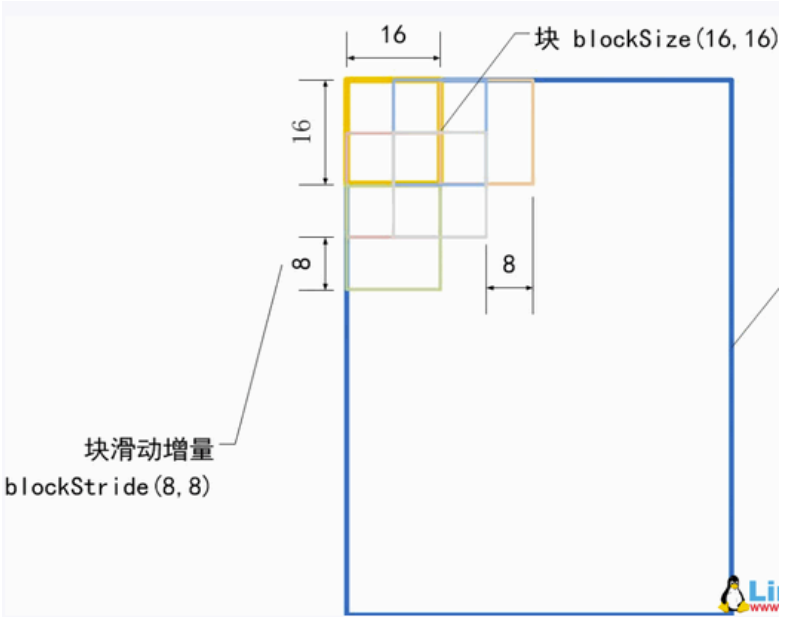
Gradient Computation

$$G_x(x,y) = H(x+1,y) - H(x-1,y)$$
$$G_y(x,y) = H(x,y+1) - H(x,y-1)$$

Orientation histogram

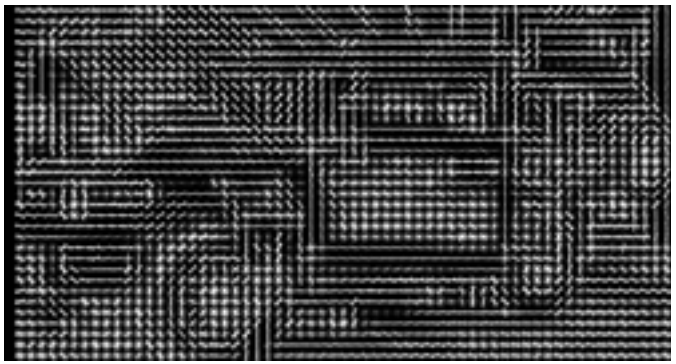






D. Dalal et al., Histogram of Oriented Gradient for Human Detection, CVPR2005

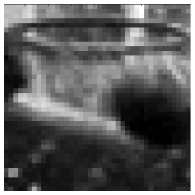
Feature: Histogram of Gradient (HoG)



- **Interesting work about HoG**

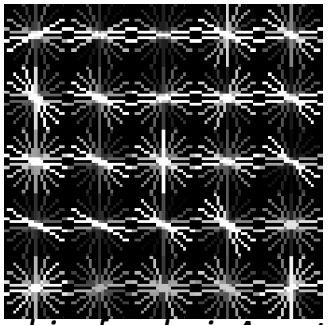
  HOGgles: Visualizing Object Detection Features, CVPR 2013

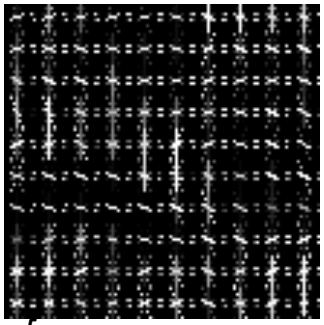
  http://web.mit.edu/vondrick/ihog



- **Nice source code, VLFeat (but only has MATLAB interfaces)**

  http://www.vlfeat.org/



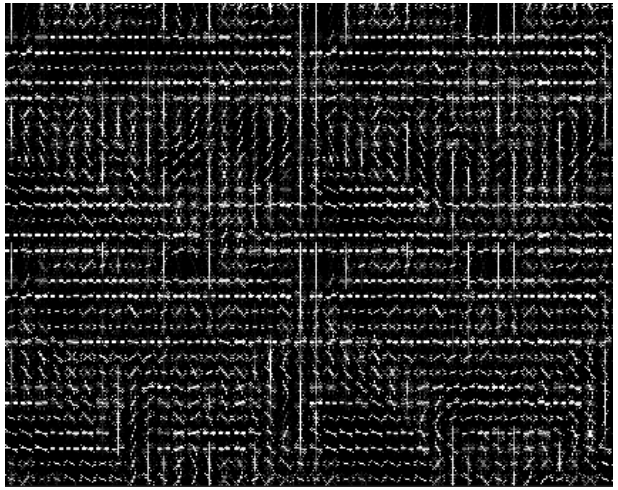


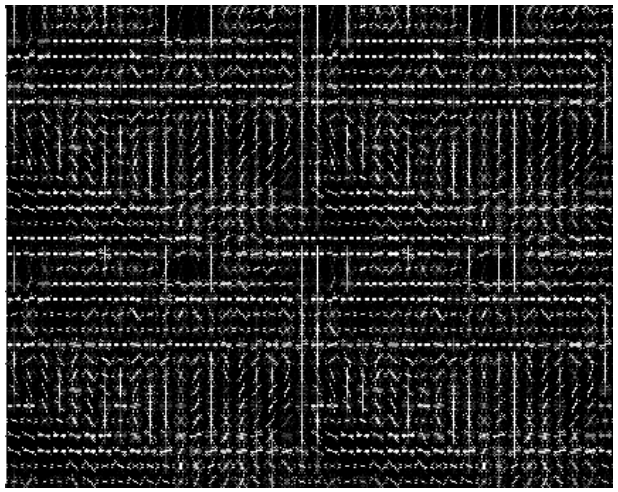
- **Use OpenCV to compute HoG**

  HOGDescriptor(_winSize, _blockSize, _blockStride, _cellSize, _nbins[, _derivAperture[, _winSigma[, _histogramNormType[, _L2HysThreshold[, _gammaCorrection[, _nlevels]]]]]])

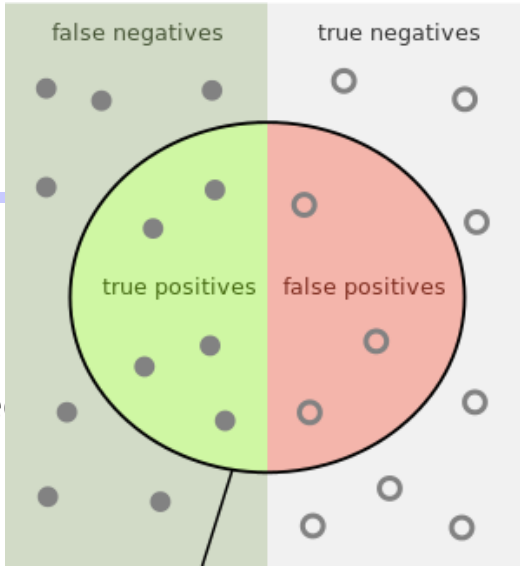
```
hog = cv2.HOGDescriptor((40, 40), (16, 16), (8, 8), (8, 8), 9)
h = hog.compute(image)
```

Feature: Histogram of Gradient (HoG)





Evaluation Criteria



In binary classification

☐ Accuracy

is the proportion of true results among the total number of cases examine

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

☐ Precision

the proportion of the true positives against all the positive results

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false positives}}$$

☐ Recall (True Positive Rate)

the proportion of the true positives against the true positives and false negatives

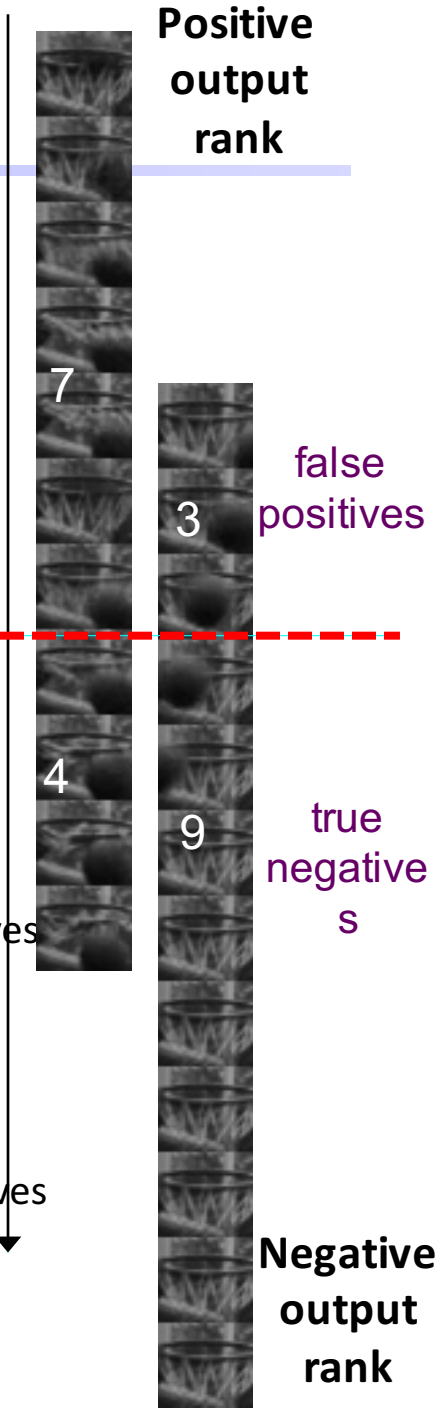
$$\text{recall} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false negatives}}$$

☐ False Alarm (False Positive Rate)

the proportion of the false positives against the false positives and true negatives

$$\text{falseAlarm rate} = \frac{\text{number of false positives}}{\text{number of false positives} + \text{true ngatives}}$$

Evaluation Criteria



In binary classification

- Accuracy

is the proportion of true results among the total number of cases examined.

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

(7+9)/(11+12)

- Precision

the proportion of the true positives against all the positive results

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false positives}}$$

7/(7+3)

- Recall (True Positive Rate)

the proportion of the true positives against the true positives and false negatives

$$\text{recall} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false negatives}}$$

7/(7+4)

- False Alarm rate (False Positive Rate)

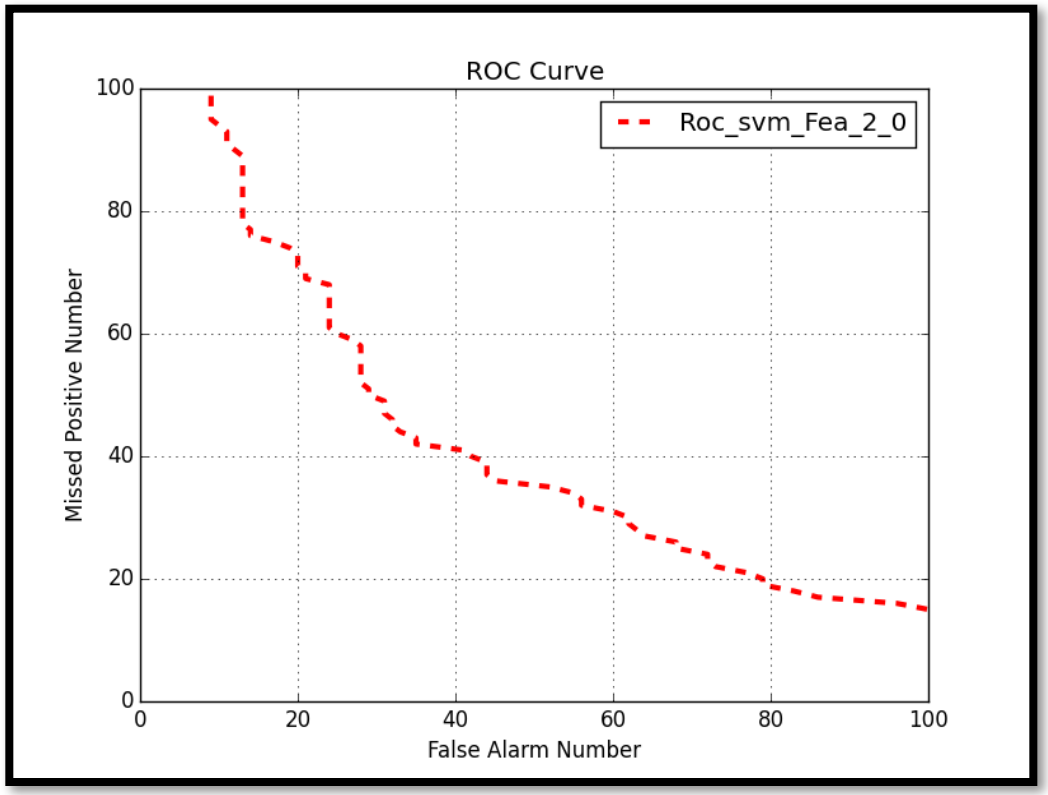
the proportion of the false positives against the false positives and true negatives

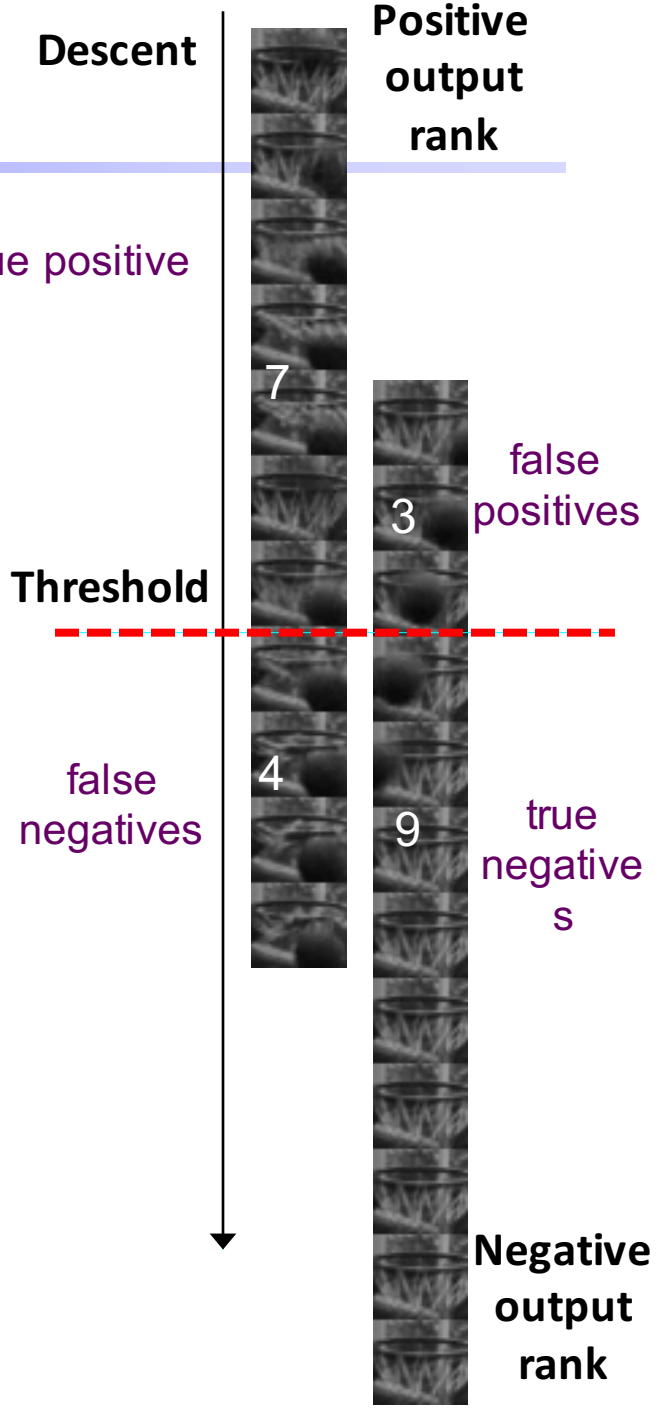
$$\text{falseAlarm rate} = \frac{\text{number of false positives}}{\text{number of false positives} + \text{true ngatives}}$$

3/(3+9)

Evaluation Criteria

- By setting different thresholds, we can obtain a sequence of (recall, false alarm rate) pairs. These can be used to plot a Receiver operating characteristic (ROC) curve.





Evaluation Criteria (a full view)

		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Test outcome	Test outcome positive	True positive	False positive (Type I error)	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Test outcome negative	False negative (Type II error)	True negative	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

Evaluation Criteria (A full view

true positive (TP)
eqv. with hit
true negative (TN)
eqv. with correct rejection
false positive (FP)
eqv. with false alarm, Type I error
false negative (FN)
eqv. with miss, Type II error

---

sensitivity or true positive rate (TPR)
eqv. with hit rate, recall
$$TPR = TP/P = TP/(TP + FN)$$
specificity (SPC) or true negative rate (TNR)
$$SPC = TN/N = TN/(FP + TN)$$
precision or positive predictive value (PPV)
$$PPV = TP/(TP + FP)$$
negative predictive value (NPV)
$$NPV = TN/(TN + FN)$$
fall-out or false positive rate (FPR)
$$FPR = FP/N = FP/(FP + TN) = 1 - SPC$$
false discovery rate (FDR)
$$FDR = FP/(FP + TP) = 1 - PPV$$
miss rate or false negative rate (FNR)
$$FNR = FN/P = FN/(FN + TP)$$

---

accuracy (ACC)
$$ACC = (TP + TN)/(P + N)$$
F1 score
is the harmonic mean of precision and sensitivity
$$F1 = 2TP/(2TP + FP + FN)$$
Matthews correlation coefficient (MCC)
$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$
Informedness = Sensitivity + Specificity - 1
Markedness = Precision + NPV - 1

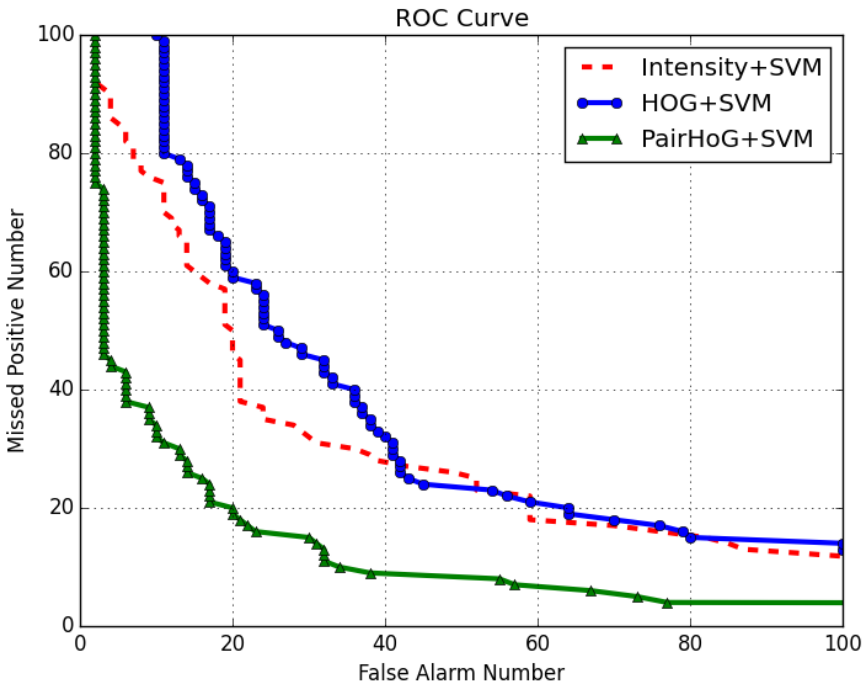
Evaluation Criteria in Practice

ROCCFG.txt

```
ROC_file_name=../ROC/Roc_svm_Fea_2_0.25_relabel.roc
ROC_file_name=../ROC/Roc_svm_Fea_2_relabel.roc
ROC_file_name=../ROC/v1/Roc_hog.roc
ROC_file_name=../ROC/v1/Roc_pairHoG.roc
ROC_file_name=../ROC/Roc_svm_Fea_2_oldAlgo_oldPartition.roc
```

```
lineStyle = ['r--', 'bo-', 'g^-', 'k-', 'rs-', 'ko-', 'r^-']
```

```
while i < len(fnROClist):
    plt.plot(rocData[i, :, 3], rocData[i, :, 2], lineStyle[mod(i, num)], linewidth=lineWidth[mod(i, num)])
    i += 1
plt.title('ROC Curve')
plt.xlabel('False Alarm Number')
plt.ylabel('Missed Positive Number')
plt.axis([0, 100, 0, 100])
plt.grid(True)
plt.legend(sMethodName)
plt.show()
```



Where are peers?

Machine Learning

- Top conferences
  - NIPS (Annual Conference on Neural Information Processing Systems)
  - ICML (International Conference on Machine Learning)
- Top Journals
  - JMLR (Journal of Machine Learning Research)
  - Neural computation (MIT Press)
  - IEEE Transactions on Neural Networks and Learning Systems

Computer Vision

- Top conferences
  - ICCV (International Conference on Computer Vision)
  - CVPR (IEEE International Conference on Computer Vision and Pattern Recognition)
  - ECCV (European Conference on Computer Vision, slightly worse)
- Top Journals
  - PAMI (IEEE Transactions on Pattern Analysis and Machine Intelligence)
  - IJCV (International Journal of Computer Computer Vision)
  - TIP (IEEE Transactions on Image Processing)

Computer Vision Resource: http://www.cvpapers.com/index.html

Some “靠谱” Websites for Q&A

Websites in English

http://stackoverflow.com/

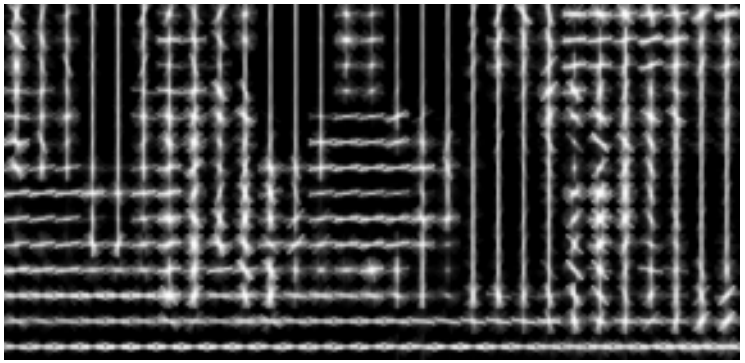
中文网站

http://www.csdn.net/

http://www.zhihu.com/topic/19559450

Assignments (only 3 days)

- To perform the project , you should team up first. We will have three teams, each with a team leader and not more than 3 members. The team leader should arrange the subtasks for all members, and finally submit the report and demo.
- Download the videos from ftp: ftp//vicstu:vicstu@10.212.48.244
- Write a annotation program to label all goal frames for one video.
- Crop the image patches to form the sample set
- Read the HoG paper to understand it, compute the HoG features, and render the HoG feature maps as follows.



- Compute the center of the positive training samples, then construct a naïve classifier by compute the distance between the testing samples and the center. Plot a ROC curve. (70% labeled samples for training and the rest for testing)
- Weekly report is required (PPT), together with code.